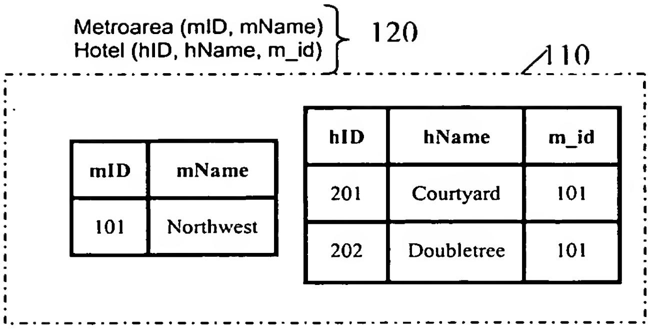


Metroarea (mID, mName)
Hotel (hID, hName, m_id)

mID	mName
101	Northwest

hID	hName	m_id
201	Courtyard	101
202	Doubletree	101

FIG. 1
PRIOR ART

200

```
<hotel>
($h = SELECT hName
FROM Hotel
)
         <metro> ($m = SELECT mName
         FROM Metroarea
         WHERE mID = $h.m_id
         )</metro>
</hotel>
```

FIG. 2
PRIOR ART

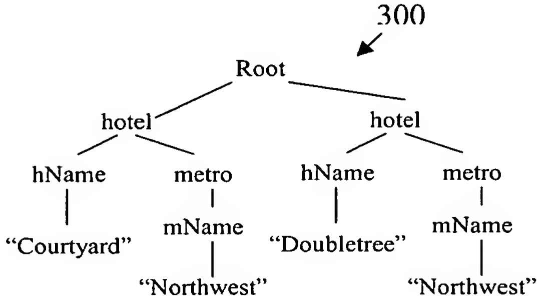
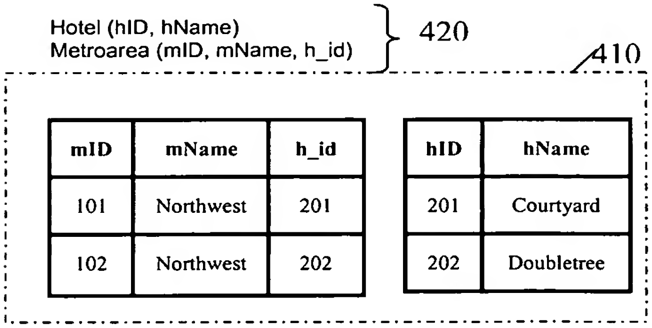


FIG. 3
PRIOR ART



Hotel (hID, hName)
Metroarea (mID, mName, h_id)

mID	mName	h_id
101	Northwest	201
102	Northwest	202

hID	hName
201	Courtyard
202	Doubletree

FIG. 4
PRIOR ART

```
Metroarea (mID, mName)
Hotel (hID, hName, m_id)
Confroom (cID, roomnum, h_id)
```

500

FIG. 5

600

```
<metro>
($m = SELECT mName
FROM Metroarea)
            <conference-room>
            ($c = SELECT cID, roomnum, m_id
            FROM Confroom, Hotel
            WHERE Confroom.h_id = Hotel.hID
                 AND Hotel.m_id = $m.mID
            )</conference-room>
</metro>
```

FIG. 6

```
Metroarea (mID, mName)
Confroom (cID, roomnum, m_id)
```

700

FIG. 7

800

```
Metroarea (mID, mName)
State (sID, sName)
Hotel (hID, hName, starrating, pool, gym, street, city, state_id,
      metro_id)
Phone (phID, phoneNo)
Confroom (cID, croomnum, capacity, rackrate, c_h_id)
Guestroom (gID, roomnum, type, rackrate, g_h_id)
Availability (aID, startdate, enddate, price, a_r_id)
Restaurant (restID, rName, rCity)
```

FIG. 8

900

```
<metro>
($m = SELECT mName FROM Metroarea)
   <hotel>
   ($h = SELECT hName, starrating, pool, gym
   FROM Hotel
   WHERE pool > 0 AND metro_id = $m.mID)
      <state>
      ($s = SELECT sName
      FROM State
      WHERE sID = $h.state_id
      )</state>

      <conference-room>
      ($c = SELECT croomnum, capacity
      FROM Confroom
      WHERE rackrate > 2 AND c_h_id = $h.hID)
            <phone-number>
            ($p = SELECT phoneNo
            FROM Phone
            WHERE phID = $h.hID
            )</phone-number>
      </conference-room>

      <guest-room>
      ($g = SELECT roomnum, type
      FROM Guestroom
      WHERE rackrate > 2 AND g_h_id = $h.hID)
            <availability>
            ($a = SELECT startdate, enddate, price
            FROM Availability
            WHERE a_r_id = $g.gID
            )</availability>
      </guest-room>

      <nearby-restaurant>
      ($r = SELECT rName, rCity
      FROM Restaurant
      WHERE rCity = $h.city
      )</nearby-restaurant>
   </hotel>
</metro>
```

FIG. 9

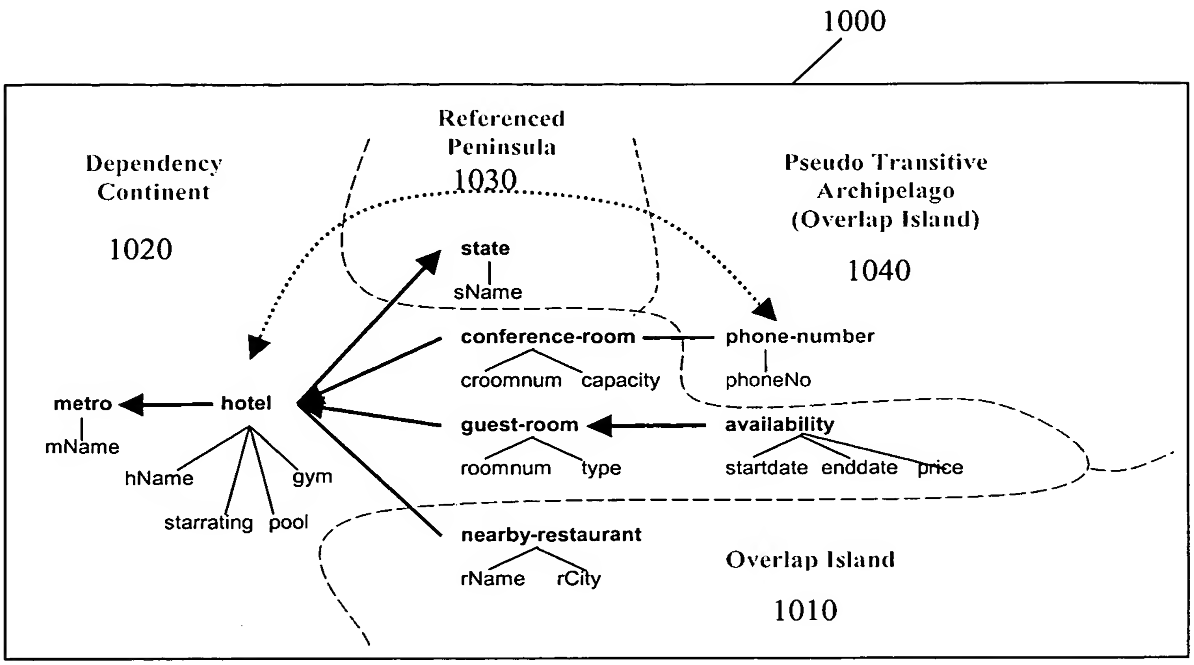
1000
Dependency Continent
1020
Referenced Peninsula
1030
Pseudo Transitive Archipelago (Overlap Island)
1040
state
sName
conference-room
croomnum
capacity
phone-number
phoneNo
metro
mName
hotel
hName
starrating
pool
gym
guest-room
roomnum
type
availability
startdate
enddate
price
nearby-restaurant
rName
rCity
Overlap Island
1010

FIG. 10

Node Categorization Process 1100

```
procedure node-cat-gen(XMLNode node)
begin
    if (node shares underlying tables with other nodes && the cardinality relationship of node and its parent is not 1:n)
    then
        node is in OI
    else
       switch (direct parent's category)
    case DC:
    switch (cardinality relationship of node and its parent)
    case 1:1:     node and its child leaf nodes are in DC
    case n:1:     node and its child leaf nodes are in DC
    case 1:n:     node and its child leaf nodes are in RP
    case m:n:     node and its child leaf nodes are in OI
    end switch
    case RP:
    if (cardinality relationship of node and its parent is m:n)
    then
       node and its child leaf nodes are in OI
    else
       node and its child leaf nodes are in RP
    case OI:
    node and its child leaf nodes are in OI
    end switch
for (each child branch node sub of node)
    node-cat-gen(sub)
end
```

FIG. 11

Deletion Translation Process 1200

```
procedure node-delete(XMLNode node)
begin
    switch (the category of node)
    case DC:
    if (node is a leaf node) then
        if (node is not a required child of its parent) then
            for the element base view of its parent, set the corresponding attribute to NULL
        else
            node cannot be deleted according to DTD
    else
        delete the corresponding tuple from element base view
        for (each child branch DC-node sub of node)
            node-delete(sub)
    case RP:
    if (node is an RP-root-node) then
        if (node is not a required child of its parent) then
            for the element base view of its parent, set the corresponding foreign key to NULL
        else
            node cannot be deleted according to DTD
    else
        node cannot be deleted to avoid side-effects
    case OI:
        node cannot be deleted to avoid side-effects
    end switch
end
```

FIG. 12

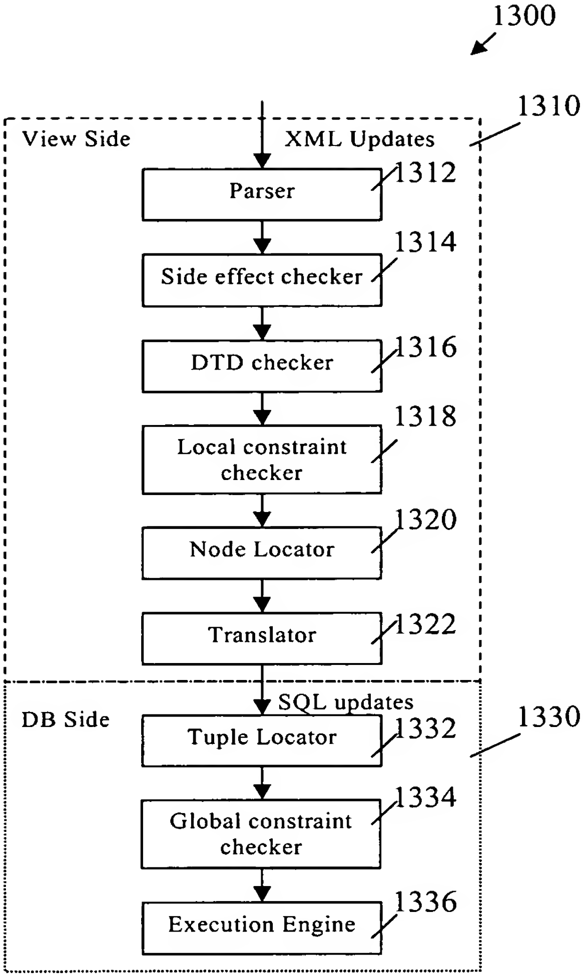

1300
1310
View Side
XML Updates
Parser
1312
Side effect checker
1314
DTD checker
1316
Local constraint checker
1318
Node Locator
1320
Translator
1322
DB Side
SQL updates
1330
Tuple Locator
1332
Global constraint checker
1334
Execution Engine
1336

FIG. 13

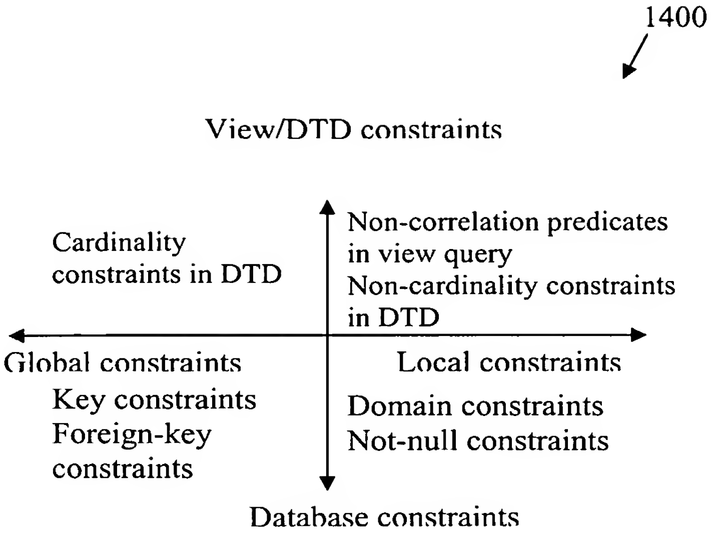

1400
View/DTD constraints
Cardinality constraints in DTD
Non-correlation predicates in view query
Non-cardinality constraints in DTD
Global constraints
Local constraints
Key constraints
Foreign-key constraints
Domain constraints
Not-null constraints
Database constraints

FIG. 14

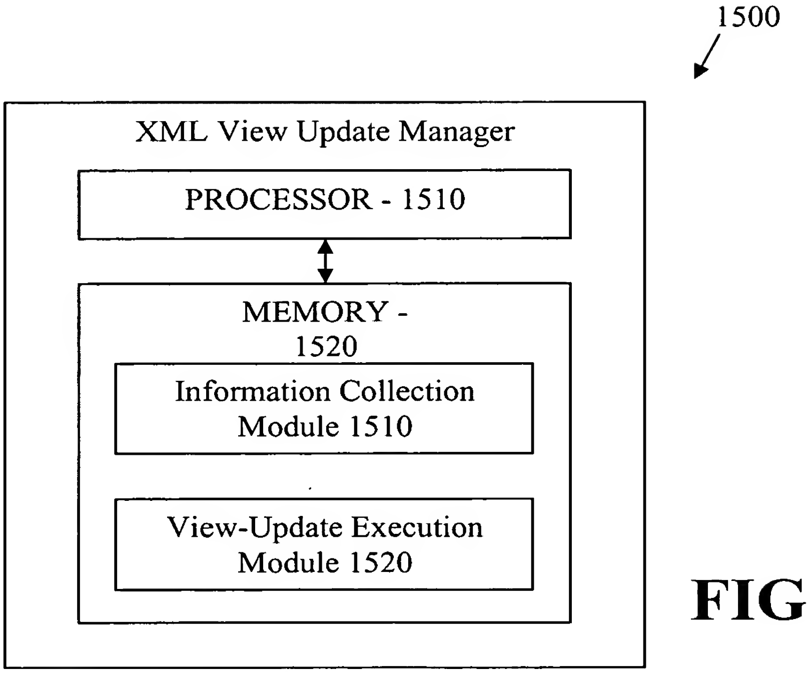

1500
XML View Update Manager
PROCESSOR - 1510
MEMORY - 1520
Information Collection Module 1510
View-Update Execution Module 1520

FIG. 15